Topics in Interoperability: Structural Programmatics in a System of Systems

James D. Smith II

October 2006

TECHNICAL NOTE
CMU/SEI-2006-TN-037

Integration of Software-Intensive Systems Initiative

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgment

The work on this technical note was partially supported by funding from the Secretary of the Air Force/Acquisition (SAF/AQ).

Abstract

This technical note presents a case study on how choices of structural programmatics (e.g., hierarchical or peer-to-peer organization, centralized or decentralized execution) affect the ability to achieve programmatic interoperability in the context of large, complex systems of systems. Key systems-of-systems concepts and definitions are introduced and explored through the case study. In addition, this report illustrates the pitfalls of focusing on only one aspect of a problem and discusses the need to balance management's desires for control with the realities of systems-of-systems programmatics. This report also introduces an alternative to conventional program management practice that addresses the pitfalls previously identified.

1 Introduction

This report draws on some recent Carnegie Mellon® Software Engineering Institute (SEI) customer engagements as the basis for a case study on how structural programmatics decisions impact the ability to achieve programmatic interoperability in a large, complex system of systems. This section introduces several important concepts, including interoperability (and, in particular, systems-of-systems interoperability), interoperable acquisition, and the relevant differences between systems of systems and traditional (i.e., monolithic) systems. Portions of this section are taken from some recent SEI reports: *Interoperable Acquisition for Systems of Systems: The Challenges* (CMU/SEI-2006-TN-034), *Topics in Interoperability: Infrastructure Replacement in a System of Systems* (CMU/SEI-2005-TN-031), and *Including Interoperability in the Acquisition Process* (CMU/SEI-2005-TR-004) [Smith 2006, Carney 2005a, Meyers 2005].

1.1 INTEROPERABILITY

Interoperability has traditionally been defined in an operational context (e.g., the ability of systems to exchange information). This definition is too imprecise and incomplete to describe the essential characteristics of interoperability, much less to allow one to reason about possible strategies to achieve—and maintain—interoperability. In the technical report entitled *System of Systems Interoperability (SOSI): Final Report*, Morris and associates discuss how interoperability is **not** a property of a system in isolation but is dependent on a particular context [Morris 2004]. Specifically, they define interoperability as

> *the ability of a set of communicating entities to (1) exchange specified state data and (2) operate on that state data according to specified, agreed-upon, operational semantics*

While this definition addresses the issue of context, it does not go far enough. The SOSI report further identifies three distinct—but interrelated—**aspects** that, taken together, provide a richer understanding of what is meant by interoperability. Figure 1, the SOSI model, illustrates the programmatic, constructive, and operational aspects of interoperability and their relationships within and across programs [Morris 2004, Meyers 2005].

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

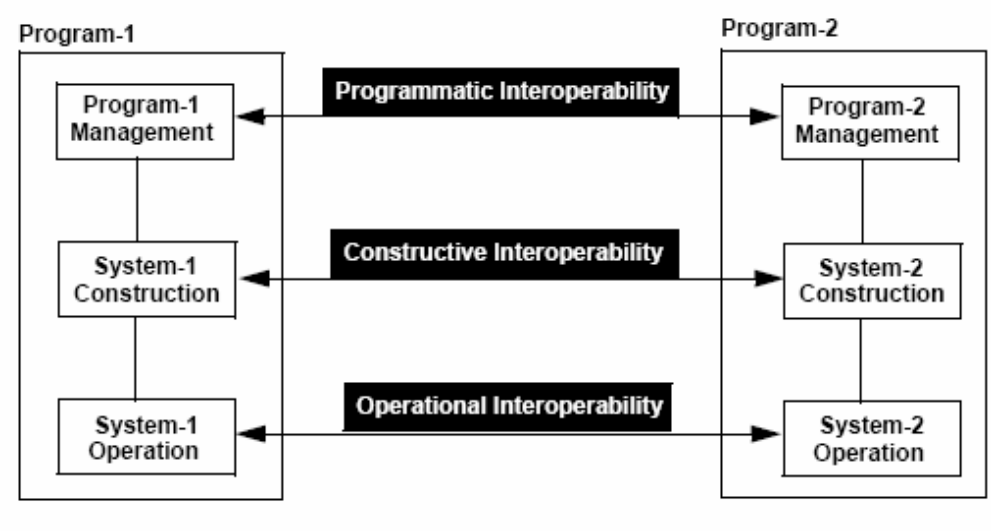

Figure 1: The SOSI Model

The three interoperability aspects are characterized as follows:

- Operational interoperability is closely aligned with the traditional definition of interoperability (the ability of systems to exchange relevant information), but it adds the notion of compatible (or complementary) operational concepts.
- Constructive interoperability reflects the degree to which the different system design, engineering, and production processes and tools are able to exchange information in an understandable manner.
- Programmatic interoperability expresses the ability of programs to exchange meaningful information about the management of the programs involved. This information can run the gamut from budget and schedule information to details on how to interpret risks.

The emphasis on **aspects** is critical: there is no such thing as a programmatic, constructive, or operational interoperability issue *per se*. Instead, there are interoperability issues that have implications or manifestations in any or (in most cases) all of the interoperability aspects. The participants in the SOSI study concluded that the impact of constructive and programmatic aspects on interoperability is significant, whereas traditional treatments of interoperability largely ignore those aspects. In fact, the participants concluded that the programmatic interoperability aspects often overwhelm the operational and constructive aspects.

1.2 SYSTEMS AND SYSTEMS OF SYSTEMS

Almost every discussion of interoperability is plagued by one annoying reality: any construct that we label a system in fact may be composed of several constituent systems, and this may be true recursively at several levels. In other words, anything that at one level we can call a system may internally be a system of systems, and any system of systems may itself be part of some larger system of {systems of systems}, and so forth.

To illustrate, we imagine some hypothetical data systems that interoperate in some manner. These data systems could all be elements (e.g., communication or navigation) of a military aircraft's

avionics system, which together with many other systems (weapons system, mission management system) compose the total aircraft, which itself can be viewed as a single system. To continue to even higher levels, the aircraft is an element in a larger system of systems, since it interoperates with other aircraft and other military units in combat. The process can continue recursively through ever larger systems of systems of systems of systems.

We illustrate this concept in Figure 2:

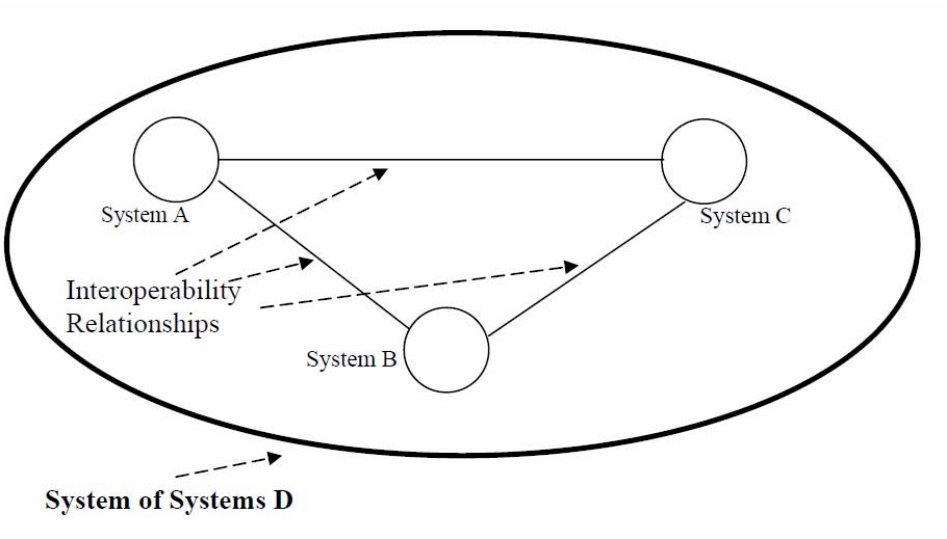

Figure 2: Systems and Systems of Systems

1.3 RELATIONSHIPS IMPLEMENTED BY SYSTEMS

A further facilitating device is that we use a common vocabulary regardless of the mechanism by which a relationship is implemented. For example, we can imagine two systems (X and Y) whose relationship is such that they must communicate data back and forth. Let us further suppose that the relationship is implemented by some complex communication system. Since that communication system is, by definition, a system in its own right, it is easy to see that discussion of such a collection may easily be complicated by two different opinions. One opinion sees a system of systems of three entities (X, Y, and the communication system). The other opinion sees a system of systems of only two (i.e., by disregarding that the communication system is a system and viewing it only as implementing the relationship between X and Y).

We argue that either view is possible, depending on the issues of immediate interest and what questions are being asked. For instance, we may be interested in the semantics of shared data between X and Y and unconcerned with the manner in which the data is communicated. In that case, we can rightly consider the communication system simply as the mechanism that implements the X-Y relationship. On the other hand, if we are concerned with the specific details of how System X locates System Y, with the significance of timing constraints and other such questions, we well may consider that the relationships between System X, System Y, and the communication system are all interoperability relationships in their own right.

1.4 SYSTEM CENTRIC VS. SYSTEM OF SYSTEMS

A system of systems has characteristics that are fundamentally different from that of a single system. These differences are summarized in Table 1.[1]

Table 1: Single System versus System of Systems

Issue	Single System	System of Systems
Constituents	All constituents are known and visible.	Changing and potentially unknown constituents • Entity assembling a system of systems may not know constituents until runtime. • Constituent may not know it is part of a system of systems.
Purpose	Predetermined by system owner and conveyed to constituents	Continuously evolving, cooperatively determined, and may or may not be known by systems participating in systems of systems
Control	Hierarchically structured with central control by system owner	System owners participating in a system of systems may have control over their systems, but they do not control how and when their systems are used in the system of systems. Entity assembling a system of systems has control over assembly but not over the participating systems.
Requirements	Defined and managed by system owner	Systems participating in the system of systems often have to anticipate how their system will be used.
Ownership	Pieces developed are owned, maintained, and evolved by system owner.	Constituent systems are independently owned, developed, maintained, and evolved.
Boundaries	Closed with clearly defined boundaries	In general, unbounded and part of larger systems of systems
Visibility	All aspects can be seen, understood, and controlled.	Components and process aspects are beyond control and visibility of developers, users, and owners.

[1] The contents of this table are taken from “Will My System ‘Play Nicely’ with Others? A Tutorial Exploring CMMI to Improve Systems of Systems Success” presented at the Software Engineering Process Group conference in 2006 (SEPG 2006) by Lisa Brownsword, Suzanne Garcia, and Grace Lewis of the SEI.

1.5 INTEROPERABLE ACQUISITION

As used in this report, **interoperable acquisition** consists of the set of practices that enable acquisition, development, and operational organizations to more effectively collaborate to field interoperable systems. This concept is distinguishable from the processes used to acquire individual systems, in that the purpose of interoperable acquisition is to influence the acquisition behavior of the constituents[2] to maximize the likelihood of a successful **system-of-systems** outcome, as opposed to maximizing the outcome for any individual system.

Why is this distinction important? Traditional system acquisition—as practiced through the ages—focuses on achieving specified performance objectives (including functional requirements, like such-and-such throughput and so-many transactions per second, as well as nonfunctional requirements, including maintainability and reliability) within cost and schedule constraints. Each system is developed in response to a perceived operational mission need and has its own operational requirements, community of interest, funding, and the like. This system-centric approach has resulted in the proliferation of stovepiped systems that are integrable with difficulty—if at all.

However, the revolution in system-of-systems thinking, as epitomized by network-centric warfare, demands a degree of collaboration and coordination among constituent systems that cannot be achieved by "adding in" something to a bunch of stovepipes (any more than maintainability, reliability, or any other desirable quality can be added-in). Instead, the entire acquisition process—from initial conception of the need for a material solution to eventual retirement of a system—must be informed by the realities of systems of systems and account for their influences. Doing so requires that the practitioner possess an understanding of the characteristics of systems of systems, as well as the different aspects of interoperability. Imparting that understanding is the focus of this report.

The rest of this report is organized as follows:

- Section 2 describes the context and background for systems-of-systems acquisition within the U.S. Department of Defense (DoD) and explores some of the pressures that influence the organizational programmatics.
- Section 3 introduces the case study and examines the approach used as well as its limitations.
- Section 4 explores some potential mitigation strategies, including an alternate approach that addresses the previously-noted limitations.
- Section 5 provides a brief conclusion.

[2] The term *constituents* encompasses all automated, mechanized, or human elements—including program offices, contractors, and systems—that have a role in a system of systems.

2 Context and Background

System acquisition, encompassing the entire life cycle—from initial formulation of a need; to program planning and budgeting; to execution, development, deployment, and operations; and through sustainment—is an inherently complex field. In the past decade, increased use of nondevelopmental components and systems—including commercial off-the-shelf (COTS), government off-the-shelf (GOTS), and free and open source software (FOSS)—forced acquirers to confront some of the limitations of systems acquisition and engineering, as traditionally practiced, and led to a new understanding of the nature of program management and systems engineering. Likewise, today's emphasis on greater interoperability and interdependence between systems, as epitomized by network-centric operations, is forcing acquirers to re-think traditional concepts about organizational programmatics and the relationship between individual programs and the larger systems of systems in which they are employed.

Just as the changes necessitated by COTS (and reuse, in general) were—and continue to be—difficult to accomplish within the cultural and regulatory context of the DoD, the clash between that context and the best practices for systems of systems is proving difficult to overcome. The reasons for this are many and include

- **no single point of ownership or control for systems of systems**
  As discussed by Carney, Anderson, and Place in *Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems*, there are many different notions of "ownership" that are applicable in a system of systems, including [Carney 2005b]
  - Ownership is not a matter of simple possession or control: There are multiple "owners" for a system of systems, representing the different acquirer, developer, user, maintainer, and other perspectives.
  - Ownership of the interoperable relationships (human-human, human-machine, and machine-machine) is not clearly understood.
  - Fundamental changes are needed in the programmatics of DoD acquisition to achieve the goals of network-centric warfare.
- **no explicit requirements or funding for systems-of-systems integration and interoperability**
  Program managers frequently complain about being required to satisfy requirements along the lines of ". . .shall be interoperable with system X"—without any characterization of what that interoperability is to be composed of and how it will be demonstrated or any explicit mention of these requirements in the documents used to prepare budget submissions, budget exhibits, or relevant appropriations [Smith 2005, Meyers 2006]. A similar condition exists today with the imposition of the Net-Ready Key Performance Parameter (NR-KPP).[3] While programs are required to incorporate the NR-KPP into their requirements, there is often no funding explicitly identified for it; so interoperability becomes another unfunded mandate.

[3] The NR-KPP has been "developed to assess net-ready attributes required for both the technical exchange of information and the end-to-end operational effectiveness of that exchange" (http://akss.dau.mil/DAG/GuideBook/IG_c7.3.4.asp).

- **no incentives for altruistic behavior on the part of program offices or system developers**
  During the course of negotiating how to achieve interoperability between systems, it often becomes apparent that changes are required of one or more systems. Changes can range from the fairly innocuous (e.g., changing the meaning of a particular bit field in a protocol exchange message) to the decidedly less so (e.g., including new message types or new protocols in a communicating system). Among a group of programs that are required to be interoperable, it may be preferable for one program rather than another to make such changes (because, for instance, the total cost—for the system of systems—would be less). If there is nothing in that program's requirements to make such changes, however, a program manager may be reluctant to do so—especially if making the changes would result in cost growth or schedule delays in the program. Such altruism for the benefit of the system of systems could be construed as "gold plating" and is not career-enhancing.
- **inappropriate processes, tools, and the like**
  As shown in the foregoing, many of the assumptions underlying traditional systems acquisition—a single point of responsibility for a system, clearly-stated requirements with associated funding, and a manager's focus on attaining cost, schedule, and performance goals—are at odds with the principles of systems of systems. Not surprisingly, the processes and tools that support acquisition are grounded in current practice and, thus, are inadequate in systems of systems. That does not mean that program managers can ignore cost, schedule, and performance in the pursuit of system-of-systems interoperability! What that **does** mean, however, is that decisions in those areas must be informed by the broader concerns of the system of systems and that additional processes and tools are necessary [Brownsword 2006].

In the next section, we will explore how these conflicting pressures combine in potentially undesirable ways—and how existing acquisition practices color what you perceive.

3 Specific Interoperability Issues

The system of systems under consideration in this case study consists of a loose confederation of systems—most of which are major defense acquisition programs in their own right—under a single program director, as shown in Figure 3. The system of systems provides a suite of capabilities to a broad range of users; the users of the system of systems depend on it to enable them to carry out their operational missions. The responsibility for developing the individual systems that compose the "core" of the system of systems belongs to the respective program offices; sustainment of that portion of the system of systems provided by the developing program offices is the responsibility of a single organization within the system of systems. The systems that form the "edge" of the system of systems are developed, acquired, deployed, and sustained by the various services' procurement and sustainment organizations.

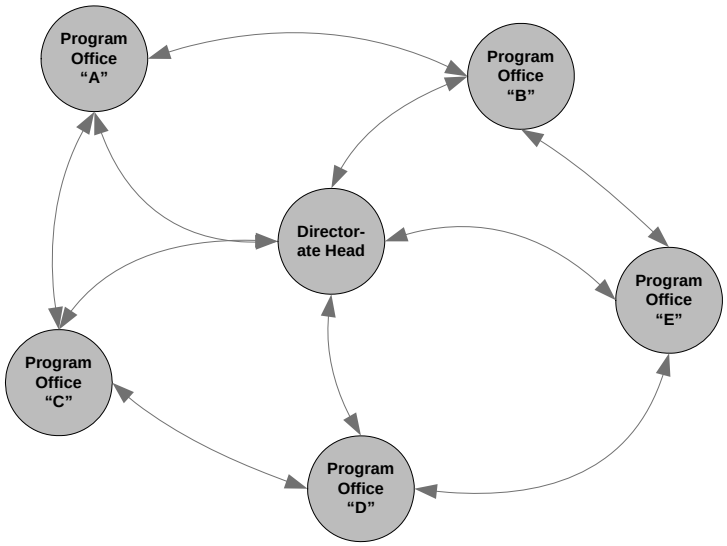

Figure 3: Federated System- of-Systems Organizational Structure

While the individual program managers have a high degree of autonomy, they aren't free to make decisions that adversely affect other portions of the system of systems. Decisions that might cause adverse effects require coordination between all the affected parties, with appropriate analysis to support the recommended decision. All of those policies are standard in large program offices or directorates.

However, despite this coordination, there had been several problems in this federated system of systems, some of which only became apparent at the "11th hour," by which point available remedies were either technically undesirable, not easily affordable, or both. When later examined, these problems were found to result mostly from ineffective communications between various system-of-systems stakeholders and senior leadership. These deficiencies resulted in senior leadership making decisions with a flawed understanding of the true state of the system of systems, based on incomplete and, often, incorrect data.

To the program director, it appeared that the root cause of the ineffective communications—and thus, the incorrect or late decisions—was the decentralized decision-making process. Thus, it was decided to centralize all programmatic decisions: individual program managers would ensure that accurate data was passed "up the chain," and the directorate office would obtain the necessary coordination with all the programs. Only then—when all the staff work and interprogram coordination was completed—would a decision be made. It was expected that this would result in decisions being coordinated, avoiding the problems that had recently plagued the system of systems. The ensuing organizational structure is shown in Figure 4.

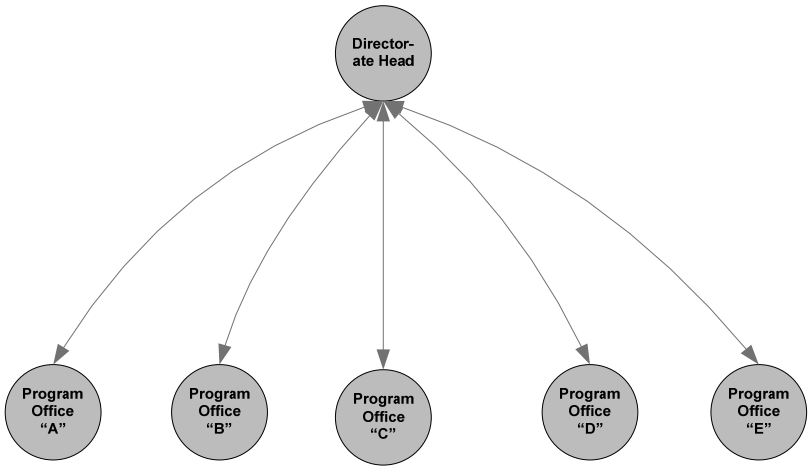

Figure 4: Hierarchical System-of-Systems Organizational Structure

While it was known that a hierarchical structure could increase the time required to pass critical information to the program director, the magnitude of this effect was unanticipated. The time required to gather, normalize, and aggregate the data and prepare the necessary documentation to support the decision-making process (through several organizational layers) was **significantly** longer than the time it took for the data to become outdated. As a consequence, by the time decisions were made, the circumstances were often so sufficiently changed that the decision was no longer correct or even relevant. So, instead of solving the problem, centralizing programmatic decisions lengthened decision timelines, while the effectiveness of interstakeholder communications remained fairly constant. The net result was, arguably, even more problems.

A couple of illustrations demonstrate this effect. One is a familiar operational analogy, the OODA Loop: Observe, Orient, Decide, and Act. A principle of U.S. military doctrine is to operate "inside" your opponent's OODA loop (i.e., to be able to see what is happening, understand the situation, and determine and implement an appropriate course of action before your opponent is able to react). The opponent in this case—composed of the system of systems and the continuously shifting state of the systems, stakeholders, and their interrelationships—was changing significantly faster than the directorate's ability to assimilate disparate information and respond accordingly.

A second example, drawing on control theory, presents a system of systems and its decision-making process as a nonlinear closed-loop feedback system.[4] The control loop time constant is the time required to gather and interpret the data and make a decision; the system-of-systems time constant is the rate at which circumstances within the system-of-systems were changing. In this case, the control loop time constant was much greater than the system-of-systems time constant, resulting in

- instability of the system of systems (i.e., decisions made so late that they tend to aggravate the situation, as opposed to mitigating issues)
- schedule delays
- cost growth
- all of the above—instability, schedule delays, and cost growth

In the case of this system of systems, the diagnosis of the root cause for the problems being experienced—ineffective communications between peers in a loose confederation—led to the formulation of a mitigation strategy intended to bring order and discipline to the decision-making process, ensuring that all decisions were properly staffed and coordinated. For a variety of reasons—including the time required to gather data, roll it up, and perform the necessary coordination prior to making any decision—problems remained. While the chosen strategy **did** give all relevant parties the opportunity to review the provided information prior to the decision—and concur or disagree—it didn't address the fundamental problem leading to the ineffective communications in the first place. Specifically, the strategy didn't address the questions of what information should be shared, how it should be shared, or how it should be used to effectively inform senior leadership. Consequently, decisions continued to be made with incomplete, incorrect, or conflicting information … but now they took longer.

[4] In engineering, control theory deals with the behavior of dynamical systems. For some background on control theory, go to http://en.wikipedia.org/wiki/Control_theory.

4 Potential Mitigation Strategies

One key to any effective system-of-systems decision process is identifying the relevant information to be shared. But what are the alternatives to achieve this? One approach could be to require that everyone share everything with everybody. While this might appeal on some "we're all in this together" level, the realities are that

- There is far too much information to share.
- Not all information needs to be—or even should be—shared.

In part, those realities are the reason why the attempt to solve the communications problems in the system of systems detailed in Section 3 failed.

An alternative approach would be based on sharing the **relevant** information with all concerned parties. This approach requires taking the time to garner an understanding of what information really needs to be shared, with whom, for what purposes, and the like—and to share only that. This is a radical departure from current practice, wherein a program manager typically decides what information another program needs. Success with this self-selection process, however, depends upon the first program manager having an accurate understanding of the information needs of the other program manager(s), as well as an understanding of the nature of the interrelationships between the programs. This level of insight is not something that can be obtained through conventional program status reviews, "stoplight charts," or integrated baseline schedules. As an example, consider the following:

> Program 1 indicates that it has a "need date" for a system (provided by Program 2) to be delivered by a certain date. This delivery is necessary so that Program 1 can complete system testing in time to meet a critical schedule milestone. Program 2 indicates that it will meet the required delivery date. Both programs are on schedule.

What's wrong with this? After all, both programs have clearly expressed their information needs and dependencies (delivery by a date certain to support a critical milestone). Unfortunately, Program 1 expected delivery to mean "installed, checkout complete, and ready for testing," but Program 2 considered it to be "sitting on the manufacturer's shipping dock." In this instance, the difference between those understandings amounted to nearly three months, forcing Program 1 to delay a critical milestone.

How did this happen? From various program reviews and quarterly information exchanges, Program 2 knew that Program 1 was concerned with the delivery date. But the understanding didn't extend beyond that. Program 2 didn't know, for instance, that Program 1 intended to commence testing the week after "delivery." Similarly, Program 1 did not understand that when Program 2 said its system was "delivered," it meant the system had been delivered to the manufacturer's shipping department for packing and shipping to Program 1. Because of this lack of understanding about the semantics of the word "delivered," neither party recognized the impending three-month program slip until very late.

Gaining an accurate understanding of these information sharing needs requires discussion—and some negotiation—with every relevant stakeholder. To do so, all parties must understand the na-

ture of the relationships that exist between them. That insight, in turn, requires a common understanding of the semantics of the relationships and information needs: what is needed, by when, for what purpose, and so on. One approach to achieving this common understanding is described in a recent SEI report entitled *System-of-Systems Navigator: An Approach for Managing System-of-Systems Interoperability* [Brownsword 2006]. Key aspects of the System-of-Systems Navigator approach include

- establishing a common understanding of the overarching goals for the system of systems as well as how every constituent contributes to the achievement of those goals (e.g., what is this "thing" supposed to do, and how does my piece contribute to the success of the whole?)
- enumerating the influence relationships between the constituents (e.g., what organizations—program offices, oversight bodies, standards groups, etc.—must my program interact with in order to be successful?)
- defining the semantics of these interrelationships (e.g., what does "delivered" mean?)
- attaining—and maintaining—agreements about these influence relationships (e.g., what are the implications to another program of a funding cut imposed on my program?)

The Navigator approach further goes on to describe some desirable characteristics of systems (or rather, system-of-systems) engineering and management processes suitable for a system of systems, as well as some considerations that need to be given to defining a rewards structure that motivates "good" system-of-systems behavior. **Note**: Navigator is not the only approach possible for dealing with the realities of systems of systems, but it does show some promise after some early pilot efforts, including one of the organizations whose experiences contributed to this case study.

5 Summary

The significant differences between traditional systems and systems of systems have profound implications to acquisition, especially when you take a broad view of acquisition. This technical note illustrates some of the pitfalls in addressing the perceived causes—instead of those actually responsible—for interoperability problems.

If, as discussed in this report, the root cause of poor decision making lies with ineffective communications, rearranging the organization chart isn't going to solve the problem. What is needed, instead, is a recognition that sharing information—an essential aspect of programmatics—

- is critical to the success of the system of systems
- needs to be treated with the same degree of seriousness afforded to creating a system architecture

Achieving this recognition requires processes to elicit an understanding of the nature of the interrelationships between the different system-of-systems constituents and involves a shared understanding of the system-of-system's objectives and constraints, as well as the role of each individual component to inform decisions about what information to share and how to share and use it. While the effort to achieve this recognition may appear unwieldy and cumbersome, its cost is considerably less than that of a wrong decision—or no decision at all.

References

[Brownsword 2006]
Brownsword, L.; Fisher, D.; Morris, E.; Smith, J.; & Kirwan, P. *System-of-Systems Navigator: An Approach for Managing System-of-Systems Interoperability* (CMU/SEI-2006-TN-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tn019.html.

[Carney 2005a]
Carney, D.; Smith, J.; & Place, P. *Topics in Interoperability: Infrastructure Replacement in a System of Systems* (CMU/SEI-2005-TN-031, ADA444901). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
http://www.sei.cmu.edu/publications/documents/05.reports/05tn031.html.

[Carney 2005b]
Carney, D.; Anderson, W.; & Place, P. *Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems* (CMU/SEI-2005-TN-046, ADA447053). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
http://www.sei.cmu.edu/publications/documents/05.reports/05tn046.html.

[Fisher 2006]
Fisher, D. *An Emergent Perspective on Interoperation in Systems of Systems* (CMU/SEI-2006-TR-003). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tr003.html.

[Meyers 2005]
Meyers, C.; Monarch, I.; Levine, L.; & Smith, J. *Including Interoperability in the Acquisition Process* (CMU/SEI-2005-TR-004, ADA441244). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
http://www.sei.cmu.edu/publications/documents/05.reports/05tr004.html.

[Meyers 2006]
Meyers, B.; Smith, J.; Capell, P.; & Place, P. *Requirements Management in a System-of-Systems Context: A Workshop* (CMU/SEI-2006-TN-015). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tn015.html.

[Morris 2004]
Morris, E.; Levine, L.; Meyers, B. C.; Place, P. R. H; & Plakosh, D. *System of Systems Interoperability (SOSI): Final Report,* (CMU/SEI-2004-TR-004, ADA455619). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/publications/documents/04.reports/04tr004.html.

[Smith 2005]
Smith II, J. & Meyers, B. *Exploring Programmatic Interoperability: Army Future Force Workshop* (CMU/SEI-2005-TN-042, ADA443482). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.
http://www.sei.cmu.edu/publications/documents/05.reports/05tn042.html.

[Smith 2006]
Smith, J. & Phillips, M. *Interoperable Acquisition for Systems of Systems: The Challenges* (CMU/SEI-2006-TN-034). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tn034.html.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2006	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Topics in Interoperability: Structural Programmatics in a System of Systems	FA8721-05-C-0003

6. AUTHOR(S)

James D. Smith II

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2006-TN-037

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note presents a case study on how choices of structural programmatics (e.g., hierarchical or peer-to-peer organization, centralized or decentralized execution) affect the ability to achieve programmatic interoperability in the context of large, complex systems of systems. Key systems-of-systems concepts and definitions are introduced and explored through the case study. In addition, this report illustrates the pitfalls of focusing on only one aspect of a problem and discusses the need to balance management's desires for control with the realities of systems-of-systems programmatics. This report also introduces an alternative to conventional program management practice that addresses the pitfalls previously identified.

14. SUBJECT TERMS	15. NUMBER OF PAGES
System, systems of systems, system of systems, programmatics, interoperability, organization	28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102